Tabela de Instruções MIPS64

Instrução	Exemplo	Significado
Add	dadd R1, R2, R3	R1 = R2 + R3
Add Immediate	daddi R1, R2, #100	R1 = R2 + 100
Subtract	dsub R1, R2, R3	R1 = R2 - R3
Sub Immediate	dsubi R1, R2, #100	R1 = R2 - 100
And Logical Immediate	andi R1, R2, #20	R1 = R2 AND 20
Or Logical	or R1, R2, R3	R1 = R2 OR R3
Load Double	ld R1, 100(R2)	R1 = Memória [R2+ 100] (lê 8 bytes)
Store Double	sd R1, 100(R2),	Memória [R2+ 100] = r1 (escreve 8 bytes)
Load Word	lw R1, 100(R2)	R1 = Memória [R2+ 100] (lê 4 bytes)
Store Word	sw R1, 100(R2)	Memória [R2+ 100] = r1 (escreve 4 bytes)
Load Byte	lb R1, 200(R2)	R1 = Memória [R2+ 200] (lê 1 byte)
Store Byte	sb R1, 200(R2)	Memória [R2+ 200] = r1 (escreve 1 byte)
Doubleword Shift Left Logical	dsll R1, R2, #2	R1 = R2 << 2 (R1 = R2 * 4) (64 bits)
Doubleword Shift Right Logical	dsrl R1, R2, #4	R1 = R2 >> 4 (R1 = R2 / 16) (64 bits)
Doubleword Shift Left Logical Variable	dsllv R1, R2, R3	R1 = R2 << R3 (R1 = R2 * 4) (64 bits)
Set Less Than	slt R1, R2, R3	Se (R2 < r3) faça R1 =1 senão R1=0
Set Less Than Immediate	slti R1, R2, #20	Se (R2 < 20) faça R1 =1 senão R1=0
Branch Equal Zero	beqz R1, endereço	Se (R1 == 0) faça PC = endereço senão PC=PC+4
Branch on Not Equal Zero	bnez R1, endereço	Se (R1 != 0) faça PC = endereço senão PC=PC+4
Branch Equal	beq R1, R2, endereço	Se (R1 == R2) faça PC = endereço senão PC=PC+4
Branch on Not Equal	bne R1, R2, endereço	Se (R1 != R2) faça PC = endereço senão PC=PC+4
Jump	j endereço	Desvia para label (PC = endereço)
Jump Register	jr R31	Desvia para R31 (PC = R31)
Jump and Link	jal endereço	R31 = PC + 4; PC = endereço (chamada de procedimento)
Jump and Link Register	jalr R20	R31 = PC +4; PC = R20 (chamada de procedimento)
Load Floating-point Register	l.d F1, 100(R1)	F1 = memória [R1 +100] (64 bits)
Store Floating-point Register	s.d F2, 100(R1)	Memória [R1 + 100] = F2 (64 bits)
Add Double Rrecision	add.d F1, F2, F3	F1 = F2+ F3 (precisão dupla)
Subtract Single Precision	sub.s F1, F2, F3	F1 = F2 – F3 (precisão simples)

Mutiply Double Precision	mul.d F1, F2, F3	F1 = F2 * F3 (precisão dupla)
Divide Single Precision	div.s F1,F2, F3	F1 = F2 / F3 (precisão simples)
Move Floating-point Register	mov.d F1, F2	F1 = F2 (precisão dupla)
Halt	halt	Pára a execução do programa
No Operation	nop	Não faz nada

Diretivas para o montador:

.data - início do segmento de dados
.text - início do segmento de programa
.code - o mesmo que .text
.org <n> - endereço inicial
.space <n> - reserva um espaço de n bytes na memória
.asciiz <s> - define uma cadeia de caracteres terminada por nulo.
.ascii <s> - define uma cadeia de caracteres
.align <n> - alinha o próximo endereço para uma fronteira de n-bytes
.word <n1>,<n2>.. - define palavras de 64 bits de dados com valor inicial
.byte <n1>,<n2>.. - define uma sequência de bytes com valor inicial.
.word32 <n1>,<n2>.. - define palavras de 32 bits com valor inicial.
.word16 <n1>,<n2>.. - define palavras de 16 bits com valor inicial.
.double <f1>,<f2>.. - define palavras no formato ponto-flutuante de 64 bits.

Onde <n> denota um número inteiro, <s> significa uma cadeia de caracteres entre aspas e <n1>, <n2> são valores inteiros separados por vírgula e <f1>, <f2> são valores reais separados por vírgula.

Outras instruções suportadas pelo simulador WinMIPS64:

daddu - add integers unsigned	daddui - add immediate unsigned
lbu - load byte unsigned	dsubu - subtract integers unsigned
lh - load 16-bit half-word	sltu - set if less than unsigned
lhu - load 16-bit half word unsigned	sltiu - set if less than or equal imm. unsigned
sh - store 16-bit half-word	dsra - shift right arithmetic
lwu - load 32-bit word unsigned	dsrlv - shift right logical by variable
lui - load upper half of register immediate	dsrav - shift right arithmetic by variable
movz - move if register equals zero	and - logical and
movn - move if register not equal to zero	xor - logical xor
cvt.w.d - convert 32-bit integer to floating-point	ori - logical or immediate
cvt.l.d - convert 64-bit integer to floating-point	xori - exclusive or immediate
mtc1 - move 32 bit integer from integer register to floating-point register	xori - logical xor immediate